# What is a machine learning problem?

A computer program is said to learn from experience E with respect to some task T and performance metric P if its performance at task T, as measured by P, improves with experience E

→ Machine learning = improving the performance P at some task T using experience E

We should define:

- improve over task T; → task T
- with respect to performance measure P; → performance metric P
- based on experience E → experience E

1

## Types of training experience

The first design choice we face is to choose the type of training experience from which our system will learn. The type of training experience available can have a significant impact on success or failure of the learner.

Attributes of the training experience:

- does the training experience provide direct feedback regarding the choices made by the performance system?
  this will lead to choose supervised/unsupervised/reinforcement learning techniques
- how well the training experience represent the distribution of examples over which the final system performance P must be measured?
  learning is most reliable when the training examples follow a distribution similar to that of future test examples; the training experience might not be fully representative of the (distribution of) situations that the learning system will face.

②

## Target function

The next design choice is to determine exactly what type of knowledge will be learned and how this will be used by the performance program.
The goal is to find an <u>operational</u> description of the real target function.

real target function $V$ ⟶ operational representation of it

An operational representation could be:

- collection of rules;
- neural network; ⟶ learning $V$ ≡ adjusting weights and biases
- polynomial function of board feature; ⟶ learning $V$ ≡ estimating coefficients
- ...

How to choose the right weights/coefficients? ⟶ learning algorithm

Generic learning algorithm

<u>Least Mean Squares (LMS)</u>: for each observed training example it adjusts the weights a small amount in the direction that reduces the error on this training example.

- initialize weights/coefficients
- for each training example b:
    - compute error(b)
        - $error(b) = V_{train}(b) - \hat{V}(b)$

          value in the train dataset of the real f.

          approximated function that the learning system is optimizing
        - if $error(b) = 0$ no weights are changed
        - if $error(b) > 0$ each weight is changed proportionally (either with respect to the importance of the corresponding feature - polynomial - or to how much the error is large (NN))

$$w_i \leftarrow w_i + \Delta \cdot error(b)$$

learning rate

# General machine learning problem

Learning a function $f: X \longrightarrow Y$, given a training set $D$ containing information about $f$.

Learning a function $f$ means computing an approximated function $\hat{f}$ that returns values as close as possible to $f$, specially for samples $x$ not present in the training set $D$.

$$\hat{f}(x) \sim f(x) \quad \forall x \in X / X_D$$

$$X_D = \{ x \mid x \in D \} \subset X \;, \quad |X_D| \ll |X|$$

We have different types of machine learning problems depending on

- type of the dataset:

  $f : X \longrightarrow Y$

  $D = \{ (x_i, y_i)_{i=1}^{n} \}$ ⟶ supervised learning

  $D = \{ (x_i)_{i=1}^{n} \}$ ⟶ unsupervised learning

  $\pi : S \longrightarrow A$

  $D = \{ (\langle s_0, a_1, r_1, s_1, \dots, a_n, r_n, s_n \rangle)_{i=1}^{n} \}$ ⟶ reinforcement learning

  ↳ learning a policy, a sequence of outputs with sparse and time delayed rewards

- type of the function to be learned:

$$\underset{\text{input domain}}{X} \equiv \begin{cases} A_1 \times \dots \times A_m, \quad A_i \text{ finite sets} \longrightarrow \text{discrete} \\ R^n \longrightarrow \text{continuous} \end{cases}$$

$$\underset{\text{output domain}}{Y} \equiv \begin{cases} R^K \longrightarrow \text{regression (approximate real-valued functions)} \\ \{ C_1, \dots, C_K \} \longrightarrow \text{classification (or pattern recognition)} \end{cases}$$

↳ if $K = 2$ we have a special case: binary classification, concept learning